
django-inline-media Documentation

Release 1.4.2

Daniel Rus Morales

Nov 07, 2017

Contents

1	Demo projects	3
1.1	Demo sites setup	3
1.2	Demo project structure	3
1.3	Example articles	4
2	Tutorial	9
2.1	Motivation	9
2.2	Installation	9
2.3	Configuration	10
2.4	Using inline-media	10
2.5	In action	11
3	Filters	13
3.1	Filter: render_inlines	13
3.2	Filter: extract_inlines	14
4	Settings	15
4.1	INLINE_MEDIA_TYPES	15
4.2	INLINE_MEDIA_CUSTOM_SIZES	15
4.3	INLINE_MEDIA_TEXTAREA_ATTRS	16
4.4	INLINE_MEDIA_REMOVE_TAGS	17
4.5	ADMIN_IMAGES_PATH	17
5	Templates	19
5.1	Template customization	20
5.2	Your own InlineTypes	20
6	Quick start	21
7	Indices and tables	23

django-inline-media allows insertion of inline media content in your text fields. Based on [django-basic-apps/inlines](#), it provides the following features:

1. Inserts pictures and collection of pictures into your texts using the `TextAreaWithInlines` widget.
2. Positions media content at different places and sizes (mini/small/medium/large at left/right or full at the left/center/right).
3. Facilitates administration with thumbnails and search by tags, author and license.
4. Shows a customised control to insert media content in text fields.
5. Uses jquery [prettyPhoto](#) to show pictures and galleries when clicking on them.
6. Tested under:
 - Python 2.7 and Python 3 (3.2, 3.4, 3.5, 3.6)
 - Django 1.8, Django 1.9 and Django 1.10

The following sample shows a centered inline picture set inserted in a text, on mouseover event the first 3 photos unfold:

Article number six

Published Feb. 16, 2012, 6:04 p.m.

Just the good ol' boys, never meanin' no harm. Beats all you've ever saw, been in trouble with the law since the day they was born. Straight'nin' the curve, flat'nin' the hills. Someday the mountain might get 'em, but the law never will. Makin' their way, the only way they know how, that's just a little bit more than the law will allow. Just good ol' boys, wouldn't change if they could, fightin' the system like a true modern day Robin Hood.



A quick walk around my 2010 in photos.
[6 pictures]

Mutley, you snickering, floppy eared hound. When courage is needed, you're never around. Those medals you wear on your moth-eaten chest should be there for bungling at which you are best. So, stop that pigeon, stop that pigeon, stop that pigeon, stop that pigeon, stop that pigeon, stop that pigeon, stop that pigeon. Howwww! Nab him, jab him, tab him, grab him, stop that pigeon now.

Run the demo project to see django-inline-media in action.

CHAPTER 1

Demo projects

There is a demo project showing stories (custom model `Article`) with a `TextFieldWithInlines`, so that the user may insert inline media content in the text.

Find the code of the example sites [here](#).

1.1 Demo sites setup

Run the demo sites in a [virtualenv](#) for this app. Create the virtualenv, clone the code and cd into any of the demo sites. Then do as follow.

```
$ cd django-inline-media/example/demo $ python manage.py syncdb --noinput $ python manage.py collectstatic $ python manage.py runserver
```

Admin user/pwd: admin/admin.

1.2 Demo project structure

The home page shows a link to an article list. The article list contains six example articles. Each of which contains pictures or picture sets located at different positions in the text. Take a look at the articles and click on the media. Pictures and picture sets are clickable by default. When clicking on a picture, the `prettyPhoto` jquery plugin overlays the picture on the current page. When clicking on a picture set, the plugin overlays a gallery view of all the pictures in the picture set.

The demo site uses **django-inline-media** with a custom **articles** app. The **articles** app defines the **Article** model. **django-inlines-media** provides 4 models: **InlineType**, **License**, **Picture** and **PictureSet**:

Django administration
Welcome, **Administrator**. [Change password](#) / [Log out](#)

Site administration

Articles		
Articles	+ Add	Change
Auth		
Groups	+ Add	Change
Users	+ Add	Change
Dress_Media		
Inline types	+ Add	Change
Licenses	+ Add	Change
Picture sets	+ Add	Change
Pictures	+ Add	Change
Sites		
Sites	+ Add	Change
Tagging		
Tagged items	+ Add	Change
Tags	+ Add	Change

Recent Actions

My Actions

- [Assorted photos of a Winter in Munich](#)
Picture set
- [Skyline of Munich](#)
Picture
- [Snowfall over the rooftops](#)
Picture
- [Leuven](#)
Picture
- [Python Programming Language](#)
Picture
- [Django Web Framework](#)
Picture
- [Windows 8](#)
Picture
- [Ubuntu](#)
Picture
- [OpenSUSE](#)
Picture
- [Red Hat](#)
Picture

The **Article** model has a **body** field of type **TextFieldWithInlines**. The field uses its own widget **TextareaWithInlines** that renders an extra control to insert inline media in the textarea. The inline media content can be placed at different positions and with different size.

Positions can be **left**, **right**, or **center**. The size can be **mini** (80px width), **small** (150px width), **medium** (200px width), **large** (250px width) and **full**. Pictures at the center are in **full size**, and picturesets in the center render at a default size of 380x280 pixels. All sizes are customizable using the setting `INLINE_MEDIA_CUSTOM_SIZES`.

1.3 Example articles

Let's see how articles in the demo site look like. Following you can see example articles one, two and five. Article views are combined with their body fields in the admin UI so that you can get an idea of how inline elements look like in the textarea and what's the effect in the final rendered article.

1.3.1 Example article one

Article one is made of four text paragraphs with a picture. The picture and its description float at the right hand side of the first paragraph.


Article number one

Published Feb. 11, 2012, 4:42 p.m.

One for all and all for one, Muskehounds are always ready. One for all and all for one, helping everybody. One for all and all for one, it's a pretty story. Sharing everything with fun, that's the way to be. One for all and all for one, Muskehounds are always ready. One for all and all for one, helping everybody. One for all and all for one, can sound pretty corny. If you've got a problem chum, think how it could be.

Ten years ago a crack commando unit was sent to prison by a military court for a crime they didn't commit. These men promptly escaped from a maximum security stockade to the Los Angeles underground. Today, still wanted by the government, they survive as soldiers of fortune. If you have a problem and no one else can help, and if you can find them, maybe you can hire the A-team.

Hong Kong Phooey, number one super guy. Hong Kong Phooey, quicker than the human eye. He's got style, a groovy style, and a car that just won't stop. When the going gets tough, he's really rough, with a Hong Kong Phooey chop (Hi-Ya!). Hong Kong Phooey, number one super guy. Hong Kong Phooey, quicker than the human eye. Hong Kong Phooey, he's fan-riffic!



ubuntu

Super-fast, easy to use and free, the Ubuntu operating system powers millions of desktops, netbooks and servers around the world. Ubuntu does everything you need it to. It'll work with your existing PC files, printers, cameras and MP3 players. And it comes with thousands of free apps.

Top preview:

Body:

```
<inline type="dress_media.picture" id="8" class="inline_large_right" /><p>One for all and all for one, Muskehounds are always ready. One for all and all for one, helping everybody. One for all and all for one, it's a pretty story. Sharing everything with fun, that's the way to be. One for all and all for one, Muskehounds are always ready. One for all and all for one, helping everybody. One for all and all for one, can sound pretty corny. If you've got a problem chum, think how it could be.</p>

<p>Ten years ago a crack commando unit was sent to prison by a military court for a crime they didn't commit. These men promptly escaped from a maximum security stockade to the Los Angeles underground. Today, still wanted by the government, they survive as soldiers of fortune. If you have a problem and no one else can help, and if you can find them, maybe you can hire the A-team.</p>
```

Inlines:

Inline type: -----

Object:

Class: Small left

Add

Insert inlines into your body by choosing an inline type, then an object, then a class.

The code highlighted in blue inserts the Ubuntu logo at the top right side of the article's text. It's been added using the control **Inlines** below the body's textarea.

The attribute *type* in the `<inline>` corresponds with the *InlineType* instance of the content_type *Picture*. The attribute *id* is the object id of the picture, and *class* represents the CSS class applied when rendering the inline with the template `templates/inline_media/inline_media_picture.html`.

1.3.2 Example article two

Yet another four paragraphs example article with two pictures, both floating at the right hand side, the first one on the first paragraph and the second on the second paragraph.

Article number two

Published Feb. 12, 2012, 4:53 p.m.

Children of the sun, see your time has just begun,
searching for your ways, through adventures every day.
Every day and night, with the condor in flight, with all
your friends in tow, you search for the Cities of Gold.
Ah-ah-ah-ah... wishing for The Cities of Gold. Ah-ah-
ah-ah-ah... some day we will find The Cities of Gold.
Do-do-do-do ah-ah-ah, do-do-do-do, Cities of Gold.
Do-do-do-do, Cities of Gold. Ah-ah-ah-ah-ah... some day
we will find The Cities of Gold.

Hong Kong Phooey, number one super guy. Hong Kong
Phooey, quicker than the human eye. He's got style, a groovy
style, and a car that just won't stop. When the going gets
tough, he's really rough, with a Hong Kong Phooey chop
(Hi-Ya!). Hong Kong Phooey, number one super guy. Hong
Kong Phooey, quicker than the human eye. Hong Kong Phooey,
he's fan-riffic!

There's a voice that keeps on calling me. Down the road, that's
where I'll always be. Every stop I make, I make a new friend.
Can't stay for long, just turn around and I'm gone again. Maybe
tomorrow, I'll want to settle down, Until tomorrow, I'll just keep moving on.

There's a voice that keeps on calling me. Down the road, that's where I'll always be. Every stop



[Python](#) is a programming language that
lets you work more quickly and integrate
your systems more effectively.



[Django](#) is a high-level Python
Web framework that
encourages rapid development
and clean, pragmatic design.

Body:

```
<inline type="dress_media.picture" id="1" class="inline_large_right" /><p>Children of the sun, see
your time has just begun, searching for your ways, through adventures every day. Every day and
night, with the condor in flight, with all your friends in tow, you search for the Cities of Gold. Ah-ah-ah-
ah-ah... wishing for The Cities of Gold. Ah-ah-ah-ah-ah... some day we will find The Cities of Gold. Do-
do-do-do ah-ah-ah, do-do-do-do, Cities of Gold. Do-do-do-do, Cities of Gold. Ah-ah-ah-ah-ah... some
day we will find The Cities of Gold.</p>

<inline type="dress_media.picture" id="2" class="inline_medium_right" /><p>Hong Kong Phooey,
number one super guy. Hong Kong Phooey, quicker than the human eye. He's got style, a groovy style,
and a car that just won't stop. When the going gets tough, he's really rough, with a Hong Kong Phooey
```

Inlines: **Inline type:** ----- **Object:**  **Class:** Small left **Add**

Insert inlines into your body by choosing an inline type, then an object, then a class.

The Python logo uses the CSS class `inline_large_right` while the Django logo uses `inline_medium_right`. Both are clickable and both contain a description with an anchor element.

The change picture view for the first image, the Python one, looks like this:


Home > Dress_media > Pictures > Python Programming Language

Change picture History

Title: Python Programming Language ☒ Show as link

Picture: **Currently:** pictures/2012/Feb/16/python-logo-master-v3-TM.png
Change: Browse...

Description: Python is a programming language
that lets you work more quickly and integrate your systems
more effectively.



click to preview

Removing the tick of the box *Show as link* avoids making the image clickable. As an alternative you can also rewrite the template `inline_media/inline_media_picture.html` using the attributes at will. Take a look at the Article 4 to see an example with an inline non-clickable picture.

1.3.3 Example article five

Three paragraphs with an inline picture set. The picture set float at the right side using the *inline_medium_right* CSS class.


Article number five

Published Feb. 15, 2012, 6:01 p.m.

Hey there where ya goin', not exactly knowin', who says you have to call just one place home. He's goin' everywhere, B.J. McKay and his best friend Bear. He just keeps on movin', ladies keep improvin', every day is better than the last. New dreams and better scenes, and best of all I don't pay property tax. Rollin' down to Dallas, who's providin' my palace, off to New Orleans or who knows where. Places new and ladies, too, I'm B.J. McKay and this is my best friend Bear.

This is my boss, Jonathan Hart, a self-made millionaire, he's quite a guy. This is Mrs H., she's gorgeous, she's one lady who knows how to take care of herself. By the way, my name is Max. I take care of both of them, which ain't easy, 'cause when they met it was MURDER!

Barnaby The Bear's my name, never call me Jack or James, I will sing my way to fame, Barnaby the Bear's my name. Birds taught me to sing, when they took me to their king, first I had to fly, in the sky so high so high, so h
tune
lique
la la
call r



Assorted photos of the Winter 2005 in Munich
[5 pictures]

Body:

```
<inline type="dress_media.pictureset" id="2" class="inline_medium_right" /><p>Hey there where ya goin', not exactly knowin', who says you have to call just one place home. He's goin' everywhere, B.J. McKay and his best friend Bear. He just keeps on movin', ladies keep improvin', every day is better than the last. New dreams and better scenes, and best of all I don't pay property tax. Rollin' down to Dallas, who's providin' my palace, off to New Orleans or who knows where. Places new and ladies, too, I'm B.J. McKay and this is my best friend Bear.</p>

<p>This is my boss, Jonathan Hart, a self-made millionaire, he's quite a guy. This is Mrs H., she's gorgeous, she's one lady who knows how to take care of herself. By the way, my name is Max. I take care of both of them, which ain't easy, 'cause when they met it was MURDER!</p>
```

Inlines: **Inline type:** **Object:** **Class:**

Insert inlines into your body by choosing an inline type, then an object, then a class.

An inline picture set has different looks:

- **As an inline:** the picture set shows only the cropped version of the cover picture.
- **On mouseover:** A cropped version of the 2/3 first pictures of the set are fanned out.
- **On click:** The picture set is overlaid in a gallery view showing complete pictures.

The overlaid gallery view of the picture set of article five:

Demo Site - Article Detail

Skyline of Munich



1/5

Skyline of Munich, from the top of the hill at the Olympia Park, 2005



[back to the](#)

Django-inline-media is a simple reusable app that allows insertion of inline media content, so far pictures and picture sets, into texts.

2.1 Motivation

Django-inline-media help your users place images or collections of images as inlines in texts.

Any application used to write text that needs to insert inline pictures is a good candidate to adopt Django-inline-media. The app will render the text with inline pictures or picture sets, and when defined as clickable pictures will be overlaid in a bigger size on top of the page.

Django-inline-media comes with two media models: Picture and PictureSet, but you can create your own inline types to support other media formats or providers ([oembed](#) based content coming soon).

This tutorial explains how to install and configure django-inline-media, how to integrate it in your web project and how to use the widget. It additionally supports the [Wysihtml5](#) rich text editor by providing a replacement for the Wysihtml5's `insertImage` command. See the [demo_wysihtml5](#) for details on this feature.

2.2 Installation

Check out the sources and add the app to your project or PYTHONPATH.

Use `git`, `pip` or `easy_install` to check out django-inline-media from [Github](#) or get a release from [PyPI](#):

1. Use **git** to clone the repository, and then install the package (read more about [git](#)):
 - `git clone git://github.com/danirus/django-inline-media.git` and
 - `python setup.py install`
2. Or use **pip** (read more about [pip](#)):
 - `Do pip install django-inline-media, or`

- Edit your project's `requirements` file and append either the [Github](#) URL or the package name `django-inline-media`, and then do `pip install -r requirements`.
3. Or use **easy_install** (read more about [easy_install](#)):
 - Do `easy_install django-inline-media`

2.3 Configuration

Follow the steps:

1. Install required apps:
 - `sorl.thumbnail`: <http://pypi.python.org/pypi/sorl-thumbnail/>
 - `tagging`: <http://pypi.python.org/pypi/tagging/>
2. Add the following entries to your `settings.py`:
 - Add `inline_media`, `sorl.thumbnail` and `tagging` to `INSTALLED_APPS`.
 - Add `THUMBNAIL_BACKEND = "inline_media.sorl_backends.AutoFormatBackend"`
 - Add `THUMBNAIL_FORMAT = "JPEG"`
 - Optionally add an extra setting to control where `django-inline-media` stores images (see [Settings](#)). It has a sane default, so don't bother to much.
3. Run management commands:
 - `python manage.py syncdb` to create `inline_media` DB entities (License, Picture, PictureSet)
 - `python manage.py collectstatic` to copy CSS and JavaScript content from `inline_media` to your project's static directory

There are extra steps when planning to use the Wyshtml5 editor. Read on the specific [ref-wysihtml5-demo](#).

2.4 Using inline-media

Using `inline-media` is pretty easy:

1. Decide which fields of your models will hold inline media content (the typical candidate: a `body` field of a `blog.Post` model)
2. Change their type from **TextField** to **TextFieldWithInlines**. This change does not affect your models' table definition, but just the way fields are rendered.
3. Change the admin class of those models and make them inherit from **AdminTextFieldWithInlinesMixin**. Fields of type **TextfieldWithInlines** will be rendered as **TextareWithInlines**

Let's see it with an example: the `Article` model.

2.4.1 Example code

The `Article` model, in the demo project, has a couple of fields of type `TextField`, `abstract` and `body`. Only the field `body` will hold inline media content. `Article` definition will look as follow:

```

from inline_media.fields import TextFieldWithInlines

class Article(models.Model):
    title = models.CharField(max_length=200)
    slug = models.SlugField(unique_for_date='publish')
    abstract = models.TextField()
    body = TextFieldWithInlines()
    publish = models.DateTimeField(default=datetime.now)

```

The ArticleAdmin class will inherit from both, **AdminTextFieldWithInlinesMixin** and Django's **ModelAdmin**:

```

from django.contrib import admin
from inline_media.admin import AdminTextFieldWithInlinesMixin
from demo.articles.models import Article

class ArticleAdmin(AdminTextFieldWithInlinesMixin, admin.ModelAdmin):
    list_display = ('title', 'publish')
    list_filter = ('publish',)
    search_fields = ('title', 'abstract', 'body')
    prepopulated_fields = {'slug': ('title',)}
    fieldsets = ((None,
                  {'fields': ('title', 'slug', 'abstract', 'body',
                              'publish',)})),)

admin.site.register(Article, ArticleAdmin)

```

2.5 In action

Look at the admin site of the demo project. Click on any of the articles and see that the **inlines** field below the **body** allows you to choose between Picture and PictureSet:

Body:

<p>Just the good ol' boys, never meanin' no harm. Beats all you've ever saw, been in trouble with the law since the day they was born. Straight'nin' the curve, flat'nin' the hills. Someday the mountain might get 'em, but the law never will. Makin' their way, the only way they know how, that's just a little bit more than the law will allow. Just good ol' boys, wouldn't change if they could, fightin' the system like a true modern day Robin Hood.</p>

<inline type="inline_media.pictureset" id="1" class="inline_full" />

<p>Mutley, you snickering, floppy eared hound. When courage is needed, you're never around. Those medals you wear on your moth-eaten chest should be there for bungling at which you are best. So,

Inlines:

Inline type: Inline_media: Picture Object: Class: Small left Add

Insert inlines

- Inline_media: Picture
- Inline_media: Pictureset

Publish:

Date: 2012-02-16 Today

Time: 18:04:03 Now

Your articles detail template (example/demo/templates/articles/article_detail.html) loads the inlines templatetag and apply the render_inlines filter to the body field:

```

{% load i18n inlines %}
...
<div class="inline_media_clearfix">

```

```
{{ object.body|render_inlines }}  
</div>
```

You can also customize inline-media templates for pictures and picture sets.

Django-inline-media comes with two filters and one tag:

- filter `render_inlines`
- filter `extract_inlines`

Load the `templatetag` module to use them in your templates:

```
{% load inlines %}
```

3.1 Filter: `render_inlines`

Renders inlines in a text by passing them through inline templates.

Syntax:

```
{{ <field>|render_inlines }}
```

Inline Syntax (singular):

```
<inline type="<app_name>.<model_name>" id="<id>" class="<cssclass>" />
```

Inline Syntax (plural):

```
<inline type="<app_name>.<model_name>" ids="<id>, <id>, <id>" />
```

An inline template will be used to render the inline. Templates will be located in the following manner:

`inline_media/<app_name>_<model_name>.html`

The template will be passed the following context:

- **object**: an object for the corresponding passed id, or
- **object_list**: a list of objects for the corresponding ids.

It would be wise to anticipate both `object_list` and `object` unless you know for sure one or the other will only be present.

Example usage:

```
{{ object.body|render_inlines }}
```

3.2 Filter: `extract_inlines`

Extract inlines from a text.

Syntax:

```
{{ <field>|extract_inlines }}
```

Example usage:

```
{% for inline in object.body|extract_inlines %}
  {% ifequal inline.content_type "inline_media.picture" %}
    {% include "inline_media/inline_media_picture.html" with object=inline.object_
    ↪class=inline.class %}
  {% endifequal %}
{% endfor %}
```

Django-inline-media recognizes four setting:

4.1 INLINE_MEDIA_TYPES

Optional

Defines the inline media types available project wide.

It defaults to:

```
INLINE_MEDIA_TYPES = ['inline_media.picture',  
                      'inline_media.pictureset']
```

4.2 INLINE_MEDIA_CUSTOM_SIZES

Optional

This setting defines custom size values for the available `INLINE_MEDIA_TYPES`. By default every inline type declared in `INLINE_MEDIA_TYPES` can be rendered in **mini**, **small**, **medium**, **large** and **full** size.

`INLINE_MEDIA_CUSTOM_SIZES` is a 2-level depth dictionary to define custom size values for each of the 5 size classes. Size classes can also be disabled.

The first level contains inline types with **app_label.model** pairs as keys. The second level contains class sizes as keys and values as geometries. When the value is just an **int**, it represents the **width** of the thumbnail. When the value is a **tuple** it represents the (**width**, **height**) of the thumbnail. The value can be `None`, what means the size won't be available for that inline type.

It defaults to:

```
INLINE_MEDIA_CUSTOM_SIZES = {
    'inline_media.picture': {
        'mini': 80,
        'small': 150,
        'medium': 200,
        'large': 250,
    },
    'inline_media.pictureset': {
        'mini': None,
        'small': (150, 150),
        'medium': (200, 200),
        'large': (250, 250),
        'full': (380, 280)
    }
}
```

See that the ‘full’ class size is not defined for the type `inline_media.picture`. That doesn’t disable it. By default the 5 class sizes are active for every inline type defined in `INLINE_MEDIA_TYPES`. The purpose of this setting is either to pass a custom size in the context to the template, or to disable a class size.

To disable the ‘small’ size for type `inline_media.pictureset` just set it to `None` in your settings module:

```
INLINE_MEDIA_CUSTOM_SIZES = {
    'inline_media.pictureset': {
        'small': None,
    }
}
```

4.3 INLINE_MEDIA_TEXTAREA_ATTRS

Optional

This setting define attributes to apply to `TextareaWithInlines` widgets.

To apply common attributes to all `TextareaWithInline` widgets use the **default** key, and define attributes and values in its dictionary (see the example below).

You can also apply rendering attributes on a per `app_label.model` and `field` basis.

In this example, every `TextFieldWithInlines` field will get the `style` attribute applied by default. Then, `abstract` and `body` fields of the `articles.article` model will get the attribute `rows` applied too. The `style` attribute defined in the **default** key can be overridden by simply defining it again for an `app_label.model/field` combination:

```
INLINE_MEDIA_TEXTAREA_ATTRS = {
    'default': {
        'style': 'font: 13px monospace',
    },
    'articles.article': {
        'abstract': {
            'rows': 5
        },
        'body': {
            'rows': 20
        }
    }
}
```

Defaults to `{ }` so that no extra attributes are applied.

4.4 `INLINE_MEDIA_REMOVE_TAGS`

Optional

This setting list all the tags that could be added by the parser ‘`html.parser`’ used with BeautifulSoup4 to render the content of `TextFieldWithInlines`. ‘`html.parser`’ is the only parser available under Python 3 at the moment.

An example:

```
INLINE_MEDIA_REMOVE_TAGS = ['</br>', '</whatever>']
```

Defaults to `['</br>']`

4.5 `ADMIN_IMAGES_PATH`

Optional

This setting establishes the path under which Django admin images may be found.

An example:

```
ADMIN_IMAGES_PATH = "%s/admin/img/admin" % STATIC_URL # Django 1.3
```

Defaults to `"%s/admin/img" % settings.STATIC_URL`, the Django 1.4 admin images path.

List of template files coming with Django-inline-media.

inline_media/inline_media.picture.mini.html

Renders an <inline> picture with any of the followin CSS classes:

- inline_mini_left or inlin_mini_right.

inline_media/inline_media.picture.default.html

Renders an <inline> picture with any of the following CSS classes:

- inline_small_left or inlin_small_right
- inline_medium_left or inlin_medium_right
- inline_large_left or inlin_large_right

inline_media/inline_media.picture.full.html Renders an <inline> picture with any of the following CSS classes:
* inline_full_left, inline_full_center, inlin_full_right.

inline_media/inline_media.pictureset.mini.html

Renders an <inline> pictureset with any of the followin CSS classes:

- inline_mini_left or inline_mini_right.

inline_media/inline_media.pictureset.default.html

Renders an <inline> pictureset with any of the following CSS classes:

- inline_small_left or inline_small_right
- inline_medium_left or inline_medium_right
- inline_large_left or inline_large_right

inline_media/inline_media.pictureset.full.html

Renders an <inline> pictureset with any of the followin CSS classes:

- inline_full_left, inline_full_center, inline_full_right.

5.1 Template customization

Django-inline-media will try to use a template matching the following pattern:

- `inline_media/<app-label>.<model>.<size>.html` Being `<size>` one of the following:
 - mini
 - small
 - medium
 - large
 - full

Note: Actual size values can be customize through the setting `INLINE_MEDIA_CUSTOM_SIZES`. See it the [Settings](#).

When django-inline-media has to render an element with a CSS class like `inline_medium_left`, it will first look for the template:

- `inline_media/<app_label>.<model>.medium.html`

And if it doesn't exist it will use the default template:

- `inline_media/<app_label>.<model>.default.html`

5.2 Your own InlineTypes

If the django-inline-media models, `Picture` and `PictureSet`, are not suitable for your project or need another ones, just create your own and bind them to the app.

Once you have your model (say `MyPicture`), declare it the setting `INLINE_MEDIA_TYPES`. Your model will then show up in the dropdown list of inline types at the bottom of your `TextFieldWithInlines` fields (like the `body` field in the `Article` model of the demo).

Then create templates to render your own media content. Name your templates after the corresponding `app_label` for your model:

- `inline_media/<my_app_label>.mypicture.<size>.html`
- `inline_media/<my_app_label>.mypicture.default.html`

CHAPTER 6

Quick start

1. Get the dependencies:

```
$ pip install -r requirements.pip
```

2. In your `settings.py`:

- Add `inline_media`, `sorl.thumbnail` and `tagging` to `INSTALLED_APPS`.
- Add `THUMBNAIL_BACKEND = "inline_media.sorl_backends.AutoFormatBackend"`
- Add `THUMBNAIL_FORMAT = "JPEG"`

3. Create a model with a field of type `TextFieldWithInlines`.

4. Create an admin class for that model by inheriting from both `inline_media.admin.AdminTextFieldWithInlinesMixin` and Django's `admin.ModelAdmin`.

5. Optionally, customise `inline_media` templates by copying them from `inline_media/templates/inline_media/` to your `inline_media/` folder in your templates directory.

5. Run `manage.py` commands: `syncdb`, `collectstatic`, `runserver`.

6. Create two *InlineType* objects, one for the *Picture* model and one for the *PictureSet* model.

7. Upload some pictures and create some picture sets.

8. Add content to the model using the field `TextFieldWithInlines` and see that you can insert inline content in the textarea. It will be rendered in the position indicated by the CSS class selected in the dropdown box.

9. Hit your app's URL!

Run the **demo** in `django-inline-media/examples/demo` to see an example.

CHAPTER 7

Indices and tables

- `genindex`
- `modindex`
- `search`

A

ADMIN_IMAGES_PATH, 17
Setting, 17

C

Code, 10
Configuration, 10
customization, 19
 INLINE_MEDIA_CUSTOM_SIZES template, 19
 template, 19

D

Demo, 3
 Project, 3
 Project Structure, 3
 Setup, 3

E

extract_inlines, 14
 Filter, 14

F

Features, 1
Filter
 extract_inlines, 14
 render_inlines, 13
Filters, 12

I

inline-media, 10
INLINE_MEDIA_CUSTOM_SIZES, 15
 Setting, 15
 template customization, 19
INLINE_MEDIA_REMOVE_TAGS, 17
 Setting, 17
INLINE_MEDIA_TEXTAREA_ATTRS, 16
 Setting, 16
INLINE_MEDIA_TYPES, 15
 Setting, 15

Installation, 9

M

Motivation, 9

P

Pair: Example
 Code, 10
Project
 Demo, 3
 Structure, Demo, 3

Q

Quick
 Start, 20

R

render_inlines, 13
 Filter, 13

S

Setting
 ADMIN_IMAGES_PATH, 17
 INLINE_MEDIA_CUSTOM_SIZES, 15
 INLINE_MEDIA_REMOVE_TAGS, 17
 INLINE_MEDIA_TEXTAREA_ATTRS, 16
 INLINE_MEDIA_TYPES, 15
Settings, 14
Setup
 Demo, 3
Start
 Quick, 20
Structure
 Demo Project, 3

T

template
 customization, 19

customization, [IN-](#)
 [LINE_MEDIA_CUSTOM_SIZES](#), [19](#)
Templates, [17](#)
Types, [11](#)